Arduino in C: See Your Microcontroller in a New Way

Written By: Chandler

TOOLS:

- Computer running Linux (1)
- USB A to B cable (1)

PARTS:

- Arduino microcontroller (1)

SUMMARY

For this project, you need to have a *Linux* computer, or else the things I describe will not work. Don't fret, though, because there are plenty of tutorials out there that will help you. Piecing this together from multiple sources has been hard, but I've been able to do it on my Arduino Duemilanove and Arduino clone. Finally, I would like to say thanks to the following sources, because they helped me the most and may help you. All of my code and setup was based on these tutorials! Thank you!

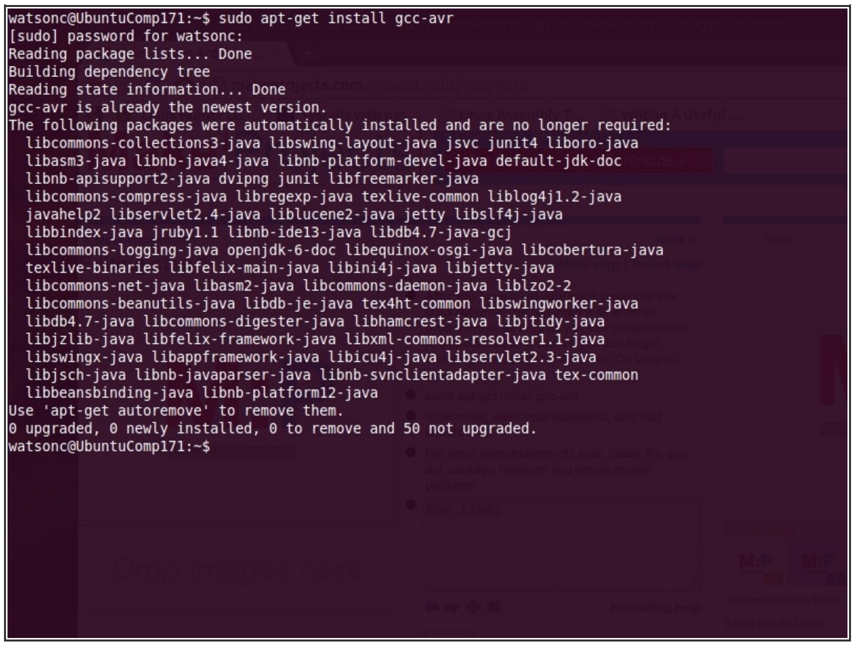
http://www.javiervalcarce.eu/wiki/Progra...

http://iamsuhasm.wordpress.com/tutsproj/...

https://www.mainframe.cx/~ckuethe/avr-c-...

http://www.micahcarrick.com/avr-tutorial...

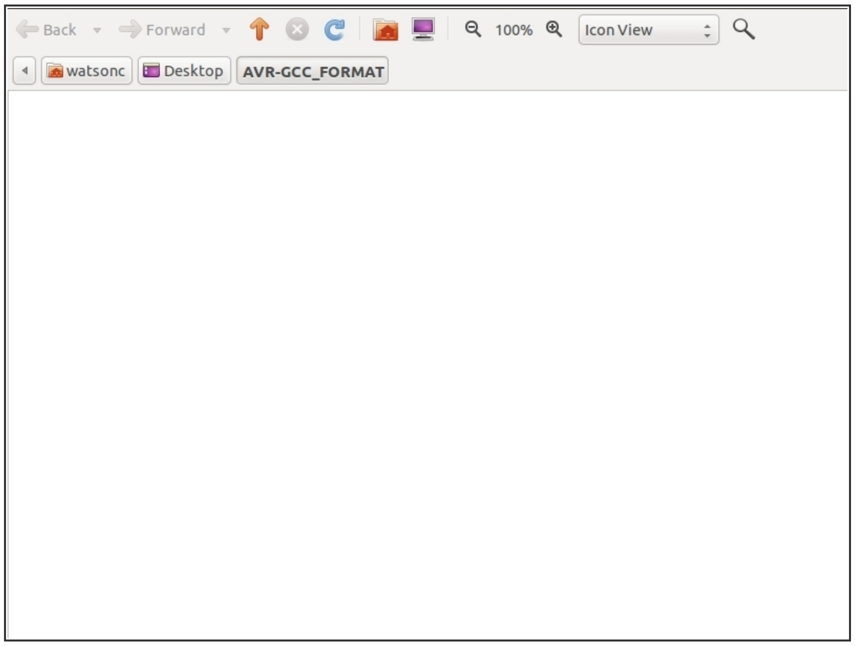
Step 1 — Arduino in C: See Your Microcontroller in a New Way



```
watsonc@UbuntuComp171:~$ sudo apt-get install gcc-avr
[sudo] password for watsonc:
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc-avr is already the newest version.
The following packages were automatically installed and are no longer required:
  libcommons-collections3-java libswing-layout-java jsvc junit4 liboro-java
  libasm3-java libnb-java4-java libnb-platform-devel-java default-jdk-doc
  libnb-apisupport2-java dvipng junit libfreemarker-java
  libcommons-compress-java libregexp-java texlive-common liblog4j1.2-java
  javahelp2 libservlet2.4-java liblucene2-java jetty libslf4j-java
  libbindex-java jruby1.1 libnb-ide13-java libdb4.7-java-gcj
  libcommons-logging-java openjdk-6-doc libequinox-osgi-java libcobertura-java
  texlive-binaries libfelix-main-java libini4j-java libjetty-java
  libcommons-net-java libasm2-java libcommons-daemon-java liblzo2-2
  libcommons-beanutils-java libdb-je-java tex4ht-common libswingworker-java
  libdb4.7-java libcommons-digester-java libhamcrest-java libjtidy-java
  libjzlib-java libfelix-framework-java libxml-commons-resolver1.1-java
  libswingx-java libappframework-java libicu4j-java libservlet2.3-java
  libjsch-java libnb-javaparser-java libnb-svnclientadapter-java tex-common
  libbeansbinding-java libnb-platform12-java
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 50 not upgraded.
watsonc@UbuntuComp171:~$
```

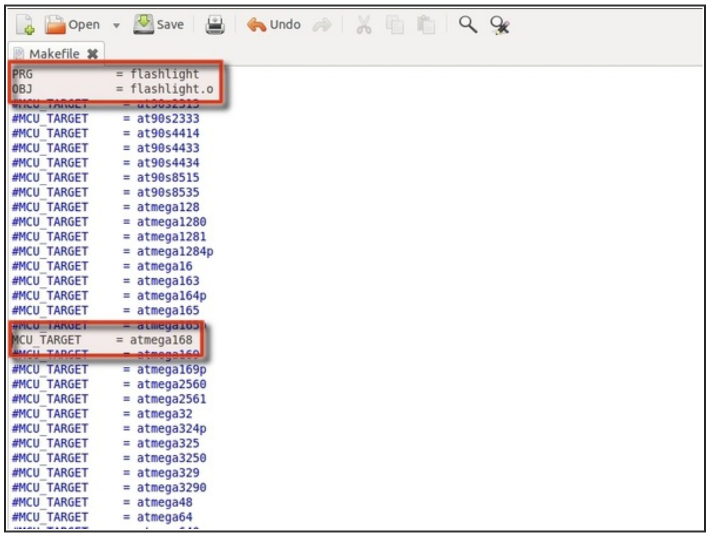
- After you have all of your parts (note that the USB cable can be replaced with another programmer, as long as you can program your Arduino or Arduino clone), you can begin installing avr-gcc (the C compiler for the Arduino). On Ubuntu, in a terminal window type:
  - `sudo apt-get install gcc-avr`
- Enter your root password, and that should do it.
- For other distrubutions of Linux, install the gcc-avr package in the same way that you would another package.

Step 2

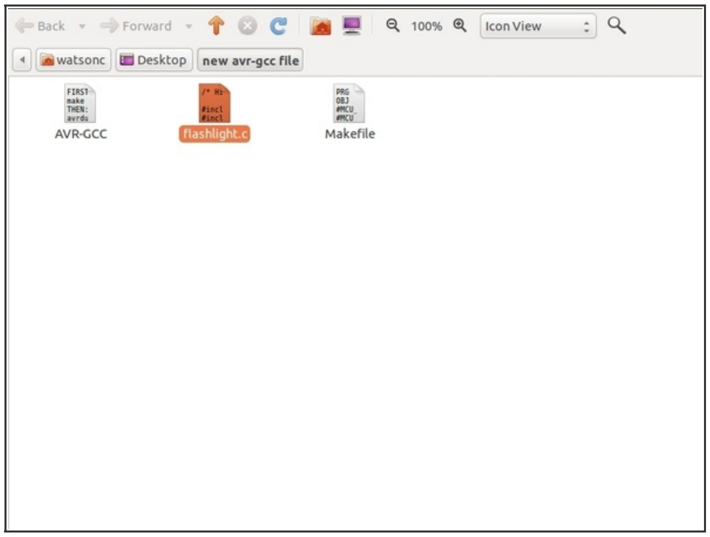


- Once that's done, you can begin. Create a new folder on your desktop called "AVR-GCC_FORMAT" or something like that. Open it and read all of the instructions. Once done with that, you can move on.

Step 3

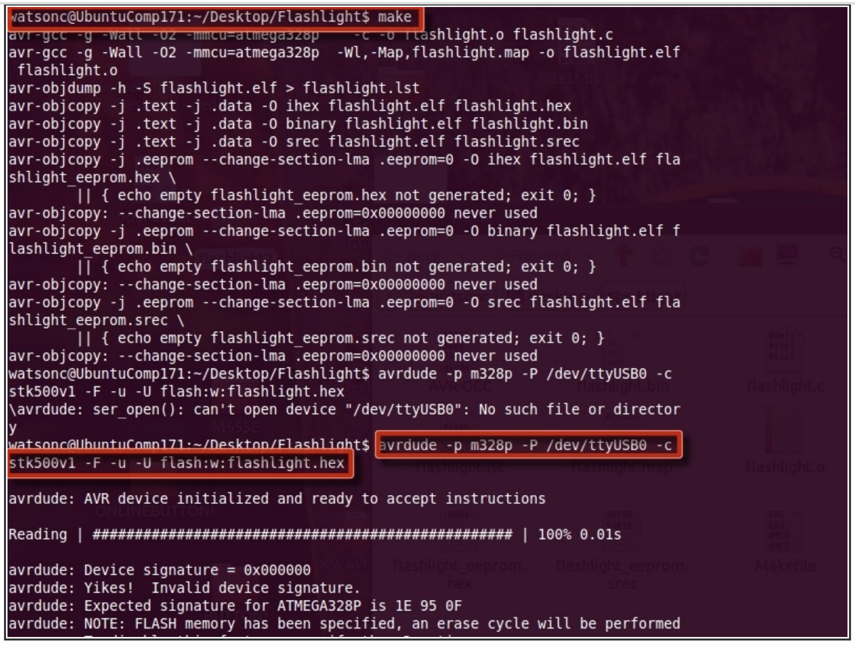


```
PRG              = flashlight
OBJ              = flashlight.o
#MCU_TARGET      = at90s2313
#MCU_TARGET      = at90s2333
#MCU_TARGET      = at90s4414
#MCU_TARGET      = at90s4433
#MCU_TARGET      = at90s4434
#MCU_TARGET      = at90s8515
#MCU_TARGET      = at90s8535
#MCU_TARGET      = atmega128
#MCU_TARGET      = atmega1280
#MCU_TARGET      = atmega1281
#MCU_TARGET      = atmega1284p
#MCU_TARGET      = atmega16
#MCU_TARGET      = atmega163
#MCU_TARGET      = atmega164p
#MCU_TARGET      = atmega165
#MCU_TARGET      = atmega165p
MCU_TARGET       = atmega168
#MCU_TARGET      = atmega169
#MCU_TARGET      = atmega169p
#MCU_TARGET      = atmega2560
#MCU_TARGET      = atmega2561
#MCU_TARGET      = atmega32
#MCU_TARGET      = atmega324p
#MCU_TARGET      = atmega325
#MCU_TARGET      = atmega3250
#MCU_TARGET      = atmega329
#MCU_TARGET      = atmega3290
#MCU_TARGET      = atmega48
#MCU_TARGET      = atmega64
```



- Now that you've done that, let's talk about what just happened. The Makefile is what compiles the "file.c" file and dispenses a .hex file, along with a bunch of other stuff. To set the makefile to your Arduino, you might have to tweak it a bit. If your Arduino is running on an ATmega328P, you're fine and can move on. If you are running any other chip (such as an ATTiny or another version of the ATmega), you will have to change the MCU_TARGET variable in the Makefile. Simply go into it, and place a # before the line:
  - `MCU_TARGET = atmega328p`
- The # denotes a comment, so you're "commenting out" that line of code. Then, find the chip that's yours (say it's an ATmega168) and remove the # before that line. Save the file, and you're done.
- Now, you can rename file.c if you want to. This is quite a dull name, after all. I'd recommend `flashlight.c`, but anything works if it ends in ".c" (and follows a bunch of weird restrictions that you must be aware of). I'd strongly recommend just to go with `flashlight.c`.
- After changing that, you have to change the Makefile's first two lines to:
  - `PRG = flashlight`
  - `OBJ = flashlight.o`
- so that the Makefile knows what it's compiling.

Step 4



```
watsonc@UbuntuComp171:~/Desktop/Flashlight$ make
avr-gcc -g -Wall -O2 -mmcu=atmega328p    -c -o flashlight.o flashlight.c
avr-gcc -g -Wall -O2 -mmcu=atmega328p  -Wl,-Map,flashlight.map -o flashlight.elf
 flashlight.o
avr-objdump -h -S flashlight.elf > flashlight.lst
avr-objcopy -j .text -j .data -O ihex flashlight.elf flashlight.hex
avr-objcopy -j .text -j .data -O binary flashlight.elf flashlight.bin
avr-objcopy -j .text -j .data -O srec flashlight.elf flashlight.srec
avr-objcopy -j .eeprom --change-section-lma .eeprom=0 -O ihex flashlight.elf fla
shlight_eeprom.hex \
        || { echo empty flashlight_eeprom.hex not generated; exit 0; }
avr-objcopy: --change-section-lma .eeprom=0x00000000 never used
avr-objcopy -j .eeprom --change-section-lma .eeprom=0 -O binary flashlight.elf f
lashlight_eeprom.bin \
        || { echo empty flashlight_eeprom.bin not generated; exit 0; }
avr-objcopy: --change-section-lma .eeprom=0x00000000 never used
avr-objcopy -j .eeprom --change-section-lma .eeprom=0 -O srec flashlight.elf fla
shlight_eeprom.srec \
        || { echo empty flashlight_eeprom.srec not generated; exit 0; }
avr-objcopy: --change-section-lma .eeprom=0x00000000 never used
watsonc@UbuntuComp171:~/Desktop/Flashlight$ avrdude -p m328p -P /dev/ttyUSB0 -c
stk500v1 -F -u -U flash:w:flashlight.hex
\avrdude: ser_open(): can't open device "/dev/ttyUSB0": No such file or director
y
watsonc@UbuntuComp171:~/Desktop/Flashlight$ avrdude -p m328p -P /dev/ttyUSB0 -c
stk500v1 -F -u -U flash:w:flashlight.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.01s

avrdude: Device signature = 0x000000
avrdude: Yikes!  Invalid device signature.
avrdude: Expected signature for ATMEGA328P is 1E 95 0F
avrdude: NOTE: FLASH memory has been specified, an erase cycle will be performed
```

- Now let's get to compiling and downloading!
- First, make a copy on the desktop of your AVR-GCC_FORMAT folder. Then, open a terminal window and `cd` to that folder. Type the command `make` and the makefile will run. A bunch of files will appear. This is why I recommended that you do this inside a folder rather than on your desktop or in your home folder.
- Now, once you're done with that, plug in your Arduino and type the following in the terminal window (make sure to reset your Arduino just before entering the command):
  - `avrdude –p` *`version`* `–P` *`location`* `–c stk500v1 –u –U –F flash:w:`*`filename`*`.hex`
- Make sure you replace *version* with the name of your chip (m328p for ATmega328P, m168 for ATmega168, or type in something random to cause an error and you'll get the rest of the list).
- *location* should be replaced with something like /dev/ttyUSB0 if that's the location of your Arduino (if you're not sure, and it's plugged into a USB port, try that first).
- Finally, *filename.hex* should be replaced with the filename, which in this case is `flashlight.hex`.

- For example, say you were downloading flashlight.hex onto an ATMega328P at /dev/ttyUSB0. You'd type: `avrdude -p m328p -P /dev/ttyUSB0 -c stk500v1 -u -U -F flash:w:flashlight.hex`

Step 5

- All the programming is done!
- Simply connect an LED from pin 13 to GND, connect a pull-down resistor (10k) from pin 4 to GND, and connect a wire to GND. Touch the other end of the wire to the part of the resistor closest to pin 4, and the light will turn on. Do it again, and it will turn off. You have completed your first project programming an Arduino in pure C!

This document was last generated on 2012-11-01 12:12:56 PM.